Version：facelib.aar

1.FACE SDK集成

- 添加三方依赖库：

```
dependencies {
    compile 'com.rokid:facelib:1.2.0.0'
}
```

- 需要如下权限：

```
网络权限：
<uses-permission android:name="android.permission.INTERNET"/>
```

读取外部存储权限： <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE/>

相机权限：

```

```

2. 接口说明及示例

```
1、单帧图片人脸检测，支持bitmap、NV21格式数据人脸检测
2、相机预览数据人脸tracking、人脸识别、人脸性别、年龄等属性识别
3、人脸数据库增删改查人脸特征值，简单实现人脸信息list序列化、反序列化存储
```

2.0 人脸检测参数配置：

2.0.1 人脸检测配置参数：

```
DFaceConf conf = new DFaceConf();
conf.setSize(width, height); // 设置数据宽高
conf.setRoi(rect); // 设置检测roi区域
conf.setDataType(type) // 设置数据格式 DataFormat

SFaceConf conf = new SFaceConf();
conf.setRecog(true, dbId); // 人脸识别开关开，dbId数据库引擎id
conf.setRecogParam(param); // TODO

DataFormat.DATA_BGR // bgr图片数据
DataFormat.DATA_BITMAP // bitmap数据
DataFormat.DATA_YUV420 // camera nv21数据
```

2.0.2 人脸数据类：

```
DataFormat.DATA_BGR // bgr图片数据
DataFormat.DATA_BITMAP // bitmap数据
DataFormat.DATA_YUV420 // camera nv21数据

FaceModel {
  List<FaceDO> faces; //人脸检测数据model，包含FaceDO list
}

FaceDO {
  public RectF faceRectF; // 人脸rect
  public long trackId;  // trackId 人脸trackId，tracking中id不变
  public byte[] UUID;   // 人脸UUID
}
```

2.0.3 人脸数据输入参数：

```
FaceInput input = new FaceInput();
input.setSize(int w, int h); // 设置输入图片数据宽高
input.setFormat(int type); // 设置图片数据格式

BitmapInput bpInput = new BitmapInput(Bitmap bp);  //支持输入bitmap
NV21Input nvInput = new NV21Input(byte[] data, int w, int h); // 支持输入
nv21数据
VideoInput videoInput = new VideoInput(byte[] data); // 支持输入相机预览数据
```

2.1 单帧图片检测：

2.1.1 人脸检测创建：

```
IImageRokidFace imageFace = ImageRokidFace.create(context);
```

- **返回：**

  `IImageRokidFace` - 单帧图片检测接口

2.1.2 人脸检测初始化：

```
imageFace.sconfig(new SFaceConf().setRecog(true, dbId); // 如果图片检测需要人
脸识别，则sconfig
```

2.1.3 人脸检测接口

```
imageFace.setImageFaceCallback(new BitmapInput(bitmap),
new ImageRokidFace.ImageFaceCallBack() {
    @Override
    public void onFaceModel(FaceModel model) {

    }});
```

- **入参：**

  `input` - `BitmapInput`类型，包含Bitmap

- **返回：**

  `model` - `FaceModel` 包含`FaceDo`类型

2.1.4 人脸检测销毁：

```
imageFace.destroy();
```

--

2.2 相机预览人脸检测：

2.2.1 人脸检测创建

```
videoFace = VideoRokidFace.create(context);
```

- **返回：**

  `IVideoRokidFace` - 相机预览数据检测接口

2.2.2 人脸检测初始化

```
VideoDFaceConf config = new VideoDFaceConf();
videoFace.dconfig(config);

SFaceConf config = new SFaceConf();
videoFace.sconfig(config.setRecog(true, dbId);
```

2.2.3 人脸检测设置相机预览数据

```
videoFace.setData(new VideoInput(bytes));
```

2.2.4 检测数据获取 byte[]

```
videoFace.getBytes();
```

2.2.5 人脸检测tracking

```
    videoFace.startTrack(new RokidFaceCallback() {
        @Override
        public void onFaceCallback(FaceModel model) {

        }
    });
```

2.2.6 人脸检测销毁：

```
videoFace.destroy();
```

--

2.3 人脸数据库操作：

2.3.1 人脸数据库DAO创建：

```
FaceDbEngine dbFace = new FaceDbEngine(context);

人脸特征库数据库，底层引擎为sqlite，人脸键值为UUID byte[32],支持add remove query操作
由于人脸特征库需要通过识别引擎提取人脸特征值后，才能做数据库add操作，需要add接口之前必须调用setData
```

- **返回：**

  ```
  `FaceDbEngine ` - 人脸数据库操作接口
  ```

2.3.2 人脸检测初始化等

```
DbEngineConfig dbConf = new DbEngineConfig(dbPath); // 设置数据库生成路径
dbFace.config(dbConf);

dbFace.getDbId(); // 获取人脸特征库dbId
```

2.3.3 人脸数据库add

```
dbFace.setData(FaceInput) //设置需要添加的人脸数据,该人脸数据必须有且只有一个人脸，否则会导致添加人脸失败
dbFace.add(UUID); // 添加UUID对应的人脸特征
```

2.3.4 人脸数据库remove

```
dbFace.remove(UUID);
```

2.3.5 人脸数据库query

```
dbFace.contain(UUID) // 是否包含UUID的特征值;
```

2.3.6 人脸数据库大小，遍历、删除

```
dbFace.dbSize() //数据库UUID数量
dbFace.getUUID(index) //根据index获取UUID
dbFace.delDb() // 删除数据库
dbFace.getDbId() //获取数据库引擎id
```

--